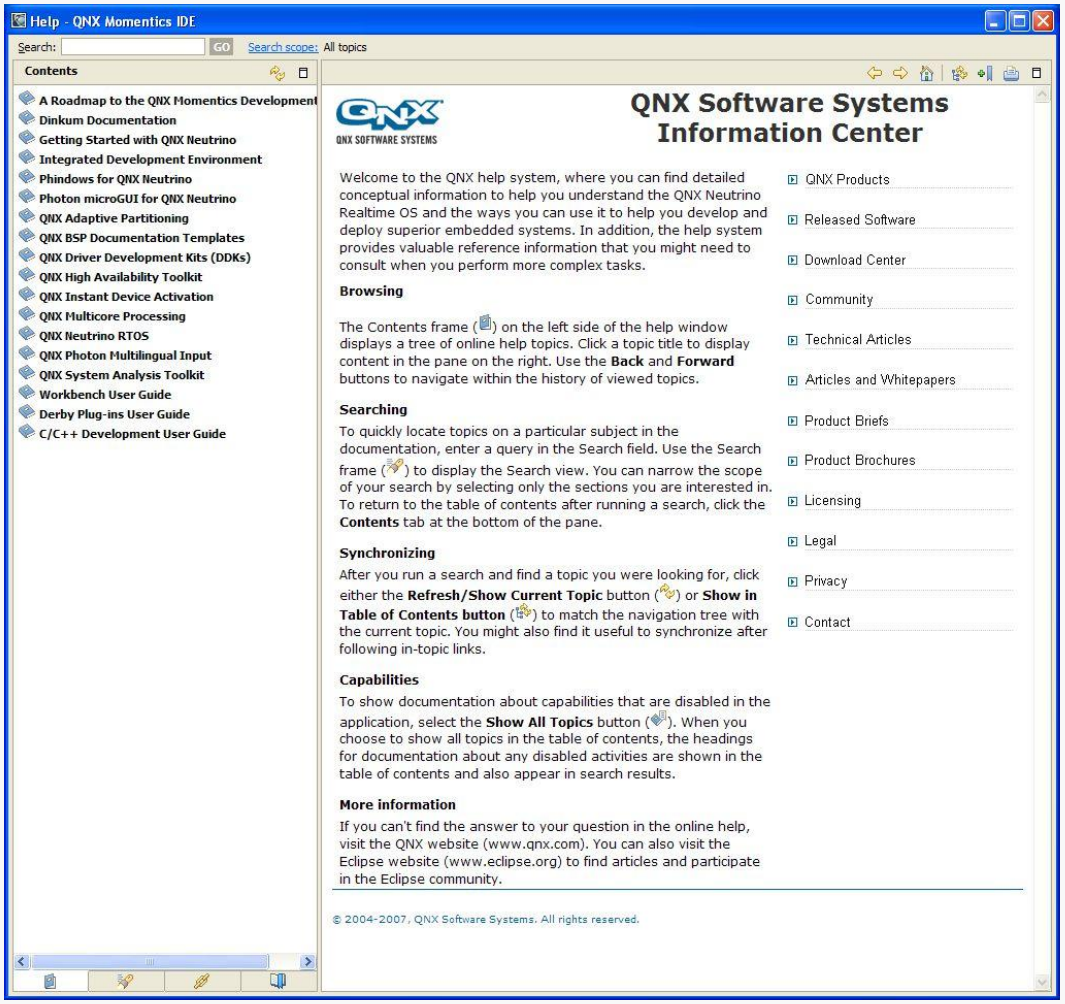
Performance Engineering of Real-Time and Embedded Systems

QNX Primitives

QNX provides some very good documentation which you get to via Help → Help Contents



- A Roadmap to the QNX Momentics Development
- Dinkum Documentation
- Getting Started with QNX Neutrino
- Integrated Development Environment
- Phindows for QNX Neutrino
- Photon microGUI for QNX Neutrino
- QNX Adaptive Partitioning
- QNX BSP Documentation Templates
- QNX Driver Development Kits (DDKs)
- QNX High Availability Toolkit
- QNX Instant Device Activation
- QNX Multicore Processing
- QNX Neutrino RTOS
- QNX Photon Multilingual Input
- QNX System Analysis Toolkit
- Workbench User Guide
- Derby Plug-ins User Guide
- C/C++ Development User Guide

QNX SOFTWARE SYSTEMS

QNX Software Systems Information Center

Welcome to the QNX help system, where you can find detailed conceptual information to help you understand the QNX Neutrino Realtime OS and the ways you can use it to help you develop and deploy superior embedded systems. In addition, the help system provides valuable reference information that you might need to consult when you perform more complex tasks.

Browsing

The Contents frame ( ) on the left side of the help window displays a tree of online help topics. Click a topic title to display content in the pane on the right. Use the **Back** and **Forward** buttons to navigate within the history of viewed topics.

Searching

To quickly locate topics on a particular subject in the documentation, enter a query in the Search field. Use the Search frame ( ) to display the Search view. You can narrow the scope of your search by selecting only the sections you are interested in. To return to the table of contents after running a search, click the **Contents** tab at the bottom of the pane.

Synchronizing

After you run a search and find a topic you were looking for, click either the **Refresh/Show Current Topic** button ( ) or **Show in Table of Contents button** ( ) to match the navigation tree with the current topic. You might also find it useful to synchronize after following in-topic links.

Capabilities

To show documentation about capabilities that are disabled in the application, select the **Show All Topics** button ( ). When you choose to show all topics in the table of contents, the headings for documentation about any disabled activities are shown in the table of contents and also appear in search results.

More information

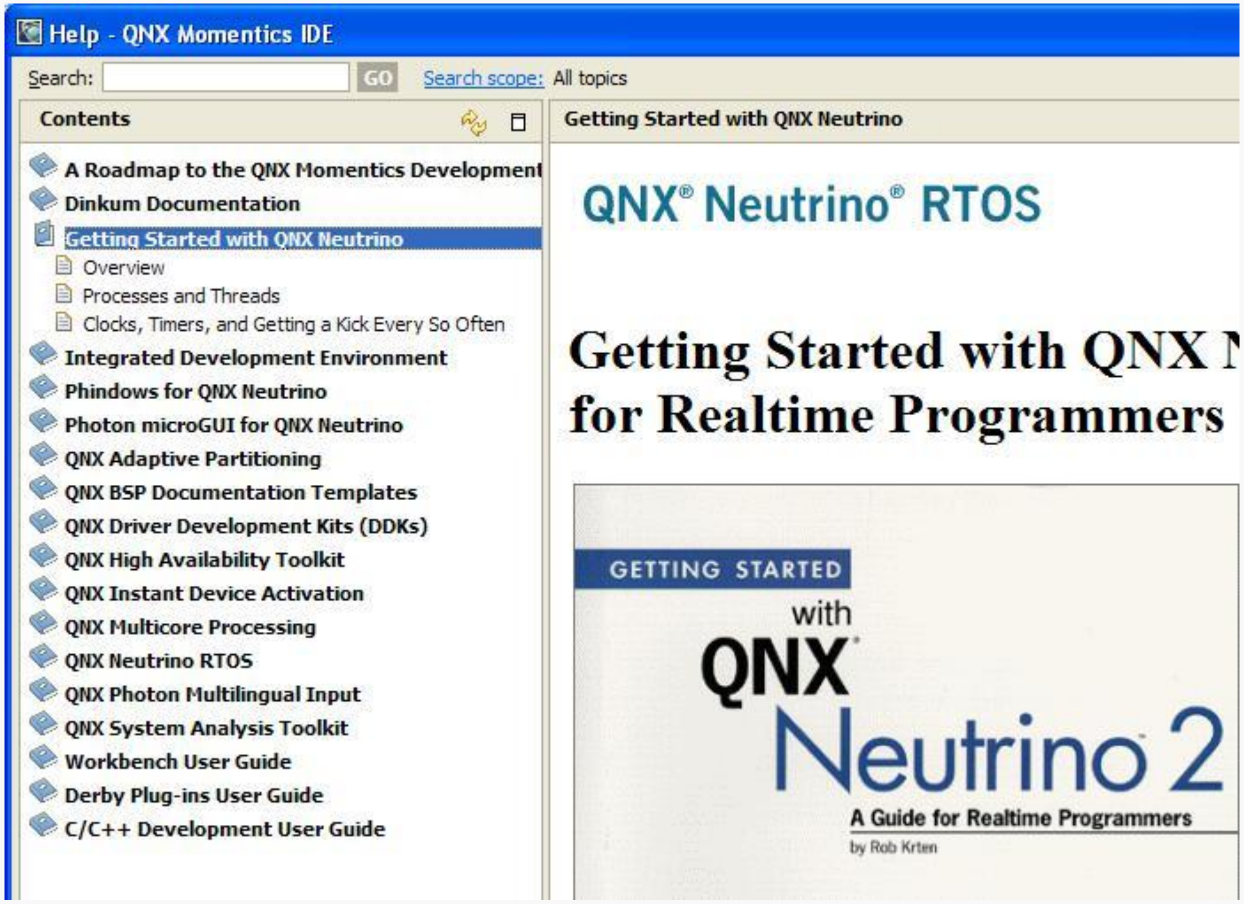
If you can't find the answer to your question in the online help, visit the QNX website (www.qnx.com). You can also visit the Eclipse website (www.eclipse.org) to find articles and participate in the Eclipse community.

- QNX Products
- Released Software
- Download Center
- Community
- Technical Articles
- Articles and Whitepapers
- Product Briefs
- Product Brochures
- Licensing
- Legal
- Privacy
- Contact

© 2004-2007, QNX Software Systems. All rights reserved.

Software Engineering

You will probably want to read the Getting Started guide which comes from the Krten book.



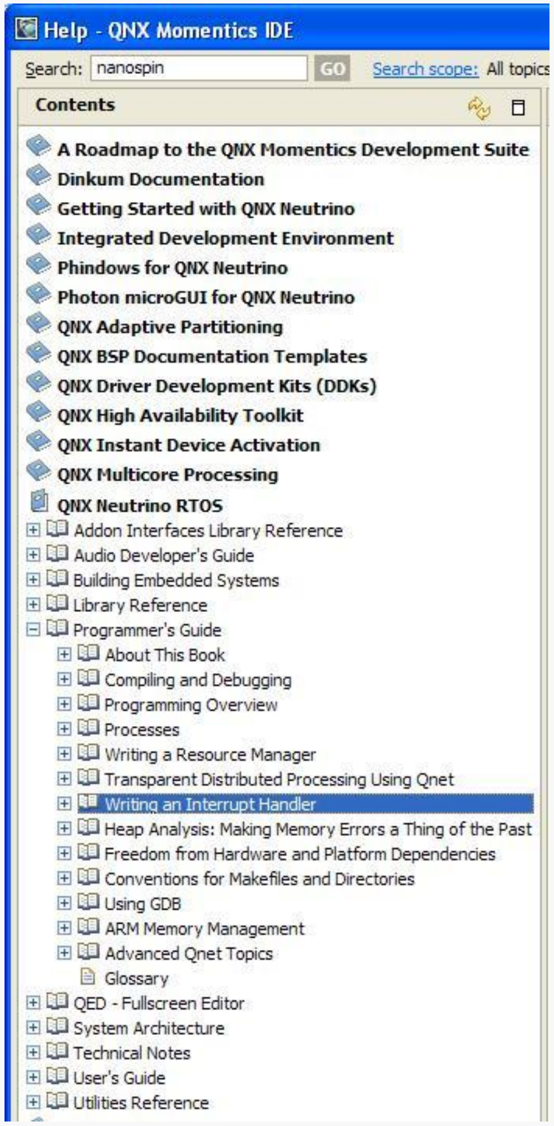
A Roadmap to the QNX Momentics Development
Dinkum Documentation
Getting Started with QNX Neutrino
Overview
Processes and Threads
Clocks, Timers, and Getting a Kick Every So Often
Integrated Development Environment
Phindows for QNX Neutrino
Photon microGUI for QNX Neutrino
QNX Adaptive Partitioning
QNX BSP Documentation Templates
QNX Driver Development Kits (DDKs)
QNX High Availability Toolkit
QNX Instant Device Activation
QNX Multicore Processing
QNX Neutrino RTOS
QNX Photon Multilingual Input
QNX System Analysis Toolkit
Workbench User Guide
Derby Plug-ins User Guide
C/C++ Development User Guide

QNX® Neutrino® RTOS

Getting Started with QNX N
for Realtime Programmers

GETTING STARTED with QNX Neutrino 2
A Guide for Realtime Programmers
by Rob Krten

If you need to write an interrupt handler look at a section in QNX Neutrino RTOS.



Help - QNX Momentics IDE

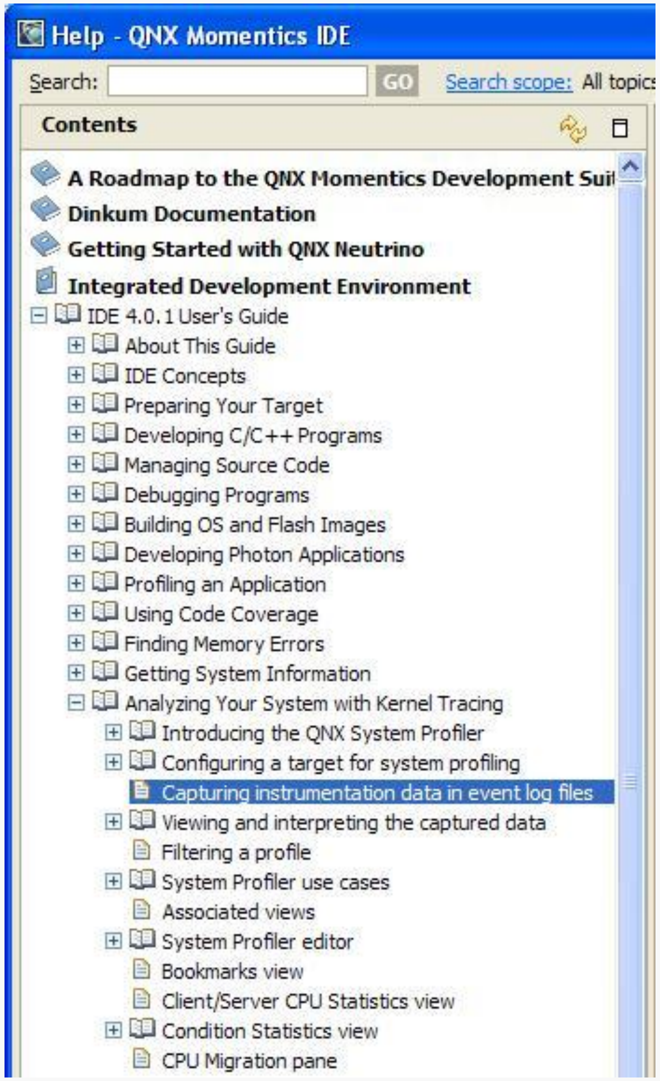
Search: nanospin GO Search scope: All topics

Contents

- A Roadmap to the QNX Momentics Development Suite
- Dinkum Documentation
- Getting Started with QNX Neutrino
- Integrated Development Environment
- Phindows for QNX Neutrino
- Photon microGUI for QNX Neutrino
- QNX Adaptive Partitioning
- QNX BSP Documentation Templates
- QNX Driver Development Kits (DDKs)
- QNX High Availability Toolkit
- QNX Instant Device Activation
- QNX Multicore Processing
- QNX Neutrino RTOS
  - Addon Interfaces Library Reference
  - Audio Developer's Guide
  - Building Embedded Systems
  - Library Reference
  - Programmer's Guide
    - About This Book
    - Compiling and Debugging
    - Programming Overview
    - Processes
    - Writing a Resource Manager
    - Transparent Distributed Processing Using Qnet
    - Writing an Interrupt Handler
    - Heap Analysis: Making Memory Errors a Thing of the Past
    - Freedom from Hardware and Platform Dependencies
    - Conventions for Makefiles and Directories
    - Using GDB
    - ARM Memory Management
    - Advanced Qnet Topics
    - Glossary
  - QED - Fullscreen Editor
  - System Architecture
  - Technical Notes
  - User's Guide
  - Utilities Reference

There is also a student tutorial available.

A section in the IDE info will help if you want to reduce the number of kernel events captured.


Help - QNX Momentics IDE

Search: GO Search scope: All topics

Contents

- A Roadmap to the QNX Momentics Development Suit
- Dinkum Documentation
- Getting Started with QNX Neutrino
- Integrated Development Environment
  - IDE 4.0.1 User's Guide
    - About This Guide
    - IDE Concepts
    - Preparing Your Target
    - Developing C/C++ Programs
    - Managing Source Code
    - Debugging Programs
    - Building OS and Flash Images
    - Developing Photon Applications
    - Profiling an Application
    - Using Code Coverage
    - Finding Memory Errors
    - Getting System Information
    - Analyzing Your System with Kernel Tracing
      - Introducing the QNX System Profiler
      - Configuring a target for system profiling
      - Capturing instrumentation data in event log files
      - Viewing and interpreting the captured data
      - Filtering a profile
      - System Profiler use cases
      - Associated views
      - System Profiler editor
      - Bookmarks view
      - Client/Server CPU Statistics view
      - Condition Statistics view
      - CPU Migration pane

QNX is a POSIX-conformant operating system.

- Any POSIX tutorial that you find on the web can provide examples and information for you.
- QNX scheduling centers around processes and threads
  - ***You build an application that runs as a process in its own address space***
  - ***Threads execute in the process' address space***
  - ***Each thread gets its own stack allocation***

pthread_create is the function you use to create a new thread

```
#include <pthread.h>

int pthread_create( pthread_t* thread,
                    const pthread_attr_t* attr,
                    void* (*start_routine)(void* ),
                    void* arg );
```

- There is a student tutorial on working with threads in an object-oriented environment.
- Naming threads is convenient for tracing

```
int pthread_setname_np(pthread_t tid,
                           const char* newname);
```

QNX provides a standard set of POSIX-conformant synchronization primitives.

- Semaphores, mutual exclusion
- Message queues
- Timers, alarms
- nano_spin – kill CPU cycles without sleeping

If you need to change task priorities for the dynamic priority scheduling algorithms…

- This primitive function will be of interest to you.

Table 2-2: **Task Scheduler Control Routines**

  - *pthread_setschedparam( )*
    Changes a number of scheduling parameters for a thread.

If you think you need to change the default timing resolution in QNX…

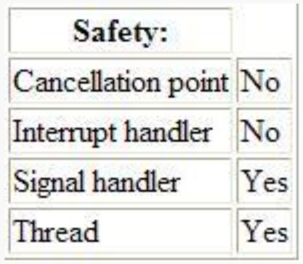
- These functions will be of interest to you.
  - *clock_getres( )* **Returns the current clock resolution.**
  - *ClockPeriod(), ClockPeriod_r()* **Get or set a clock period.**

Timers will be the way that you can wake up a thread at a specific time.

- You can setup the timers to send a pulse to you, or to send you a SIGALRM signal
  - ***The pulse you receive by checking MsgReceive***
  - ***To receive a signal you specify a callback function as the signal handler***
- There is a student tutorial available for working with timers in QNX.

When working with threads, interrupt handlers, and signal handlers be aware of safety concerns.

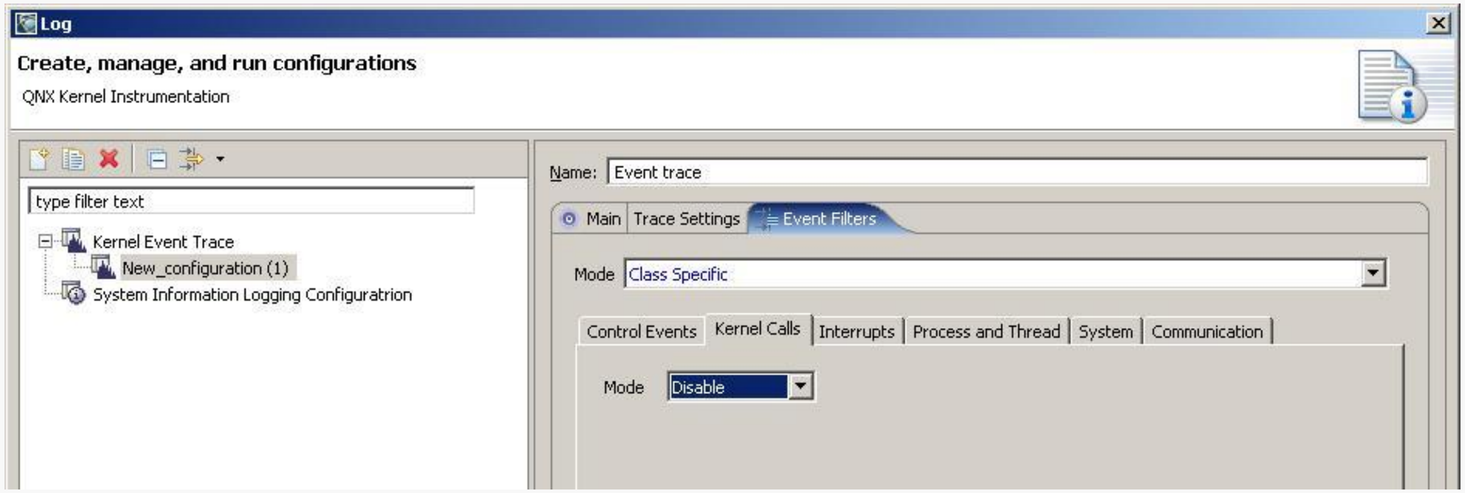
- In this case, safety is from other threads or just whether it is legal to execute an operation in that context.
- The description for each OS primitive specifies its safe usage.



Safety:	
Cancellation point	No
Interrupt handler	No
Signal handler	Yes
Thread	Yes

When you setup your kernel event logging you can control what data is captured.

Filter an entire class of events



Log

Create, manage, and run configurations

QNX Kernel Instrumentation

type filter text

- Kernel Event Trace
  - New_configuration (1)
- System Information Logging Configuratrion

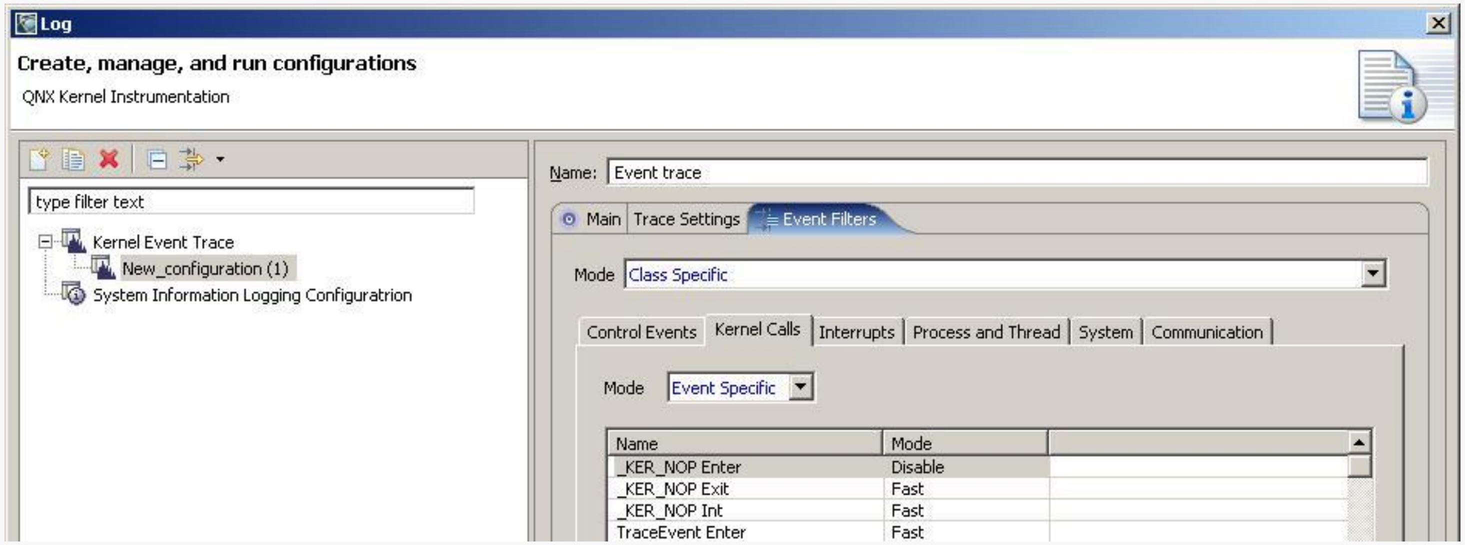
Name: Event trace

Main | Trace Settings | Event Filters

Mode: Class Specific

Control Events | Kernel Calls | Interrupts | Process and Thread | System | Communication

Mode: Disable



Log

Create, manage, and run configurations

QNX Kernel Instrumentation

type filter text

- Kernel Event Trace
  - New_configuration (1)
- System Information Logging Configuratrion

Name: Event trace

Main | Trace Settings | Event Filters

Mode: Class Specific

Control Events | Kernel Calls | Interrupts | Process and Thread | System | Communication

Mode: Event Specific

Name	Mode
_KER_NOP Enter	Disable
_KER_NOP Exit	Fast
_KER_NOP Int	Fast
TraceEvent Enter	Fast

Filter one type of event

The TraceEvent() function allows you to insert user events into the kernel event stream.

```
#include <sys/neutrino.h>
#include <sys/trace.h>

TraceEvent(_NTO_TRACE_INSERTSUSEREVENT,
           int event, int data0, int data1)
TraceEvent(_NTO_TRACE_INSERTCUSEREVENT,
           int event, unsigned * buf, unsigned len)
TraceEvent(_NTO_TRACE_INSERTUSRSTREVENT,
           int event, const char * str)
```

`event` must be between `_NTO_TRACE_USERFIRST` and `_NTO_TRACE_USERLAST`

Look at the examples in the System Analysis Toolkit guide.

You can add these statements to your application even if you set up kernel logging through Momentics.